

Webサービスのセキュリティ自己 点検 30項目

公開前・取引先提出前に、開発者自身で確認する

発行 株式会社Forest Book (Forest Book Inc.)

株式会社Forest Book Forest Security

2026年10月 版 1.0

本資料の複製・社内配布は自由。再配布時は出典を明記のこと。

表紙情報

項目	内容
資料名	Webサービスのセキュリティ自己点検 30項目 ― 公開前・取引先提出前に、開発者自身で確認する
発行	株式会社Forest Book（Forest Book Inc.）Forest Security
版	1.0（2026年10月）
想定読者	Webアプリ、API、SaaS、EC、会員サイトを自社で開発・運用する中小企業・開発会社の開発責任者、テックリード、情報システム担当
前提	特定の言語・フレームワーク・クラウドに依存しない。コード例やツール名は例示であり、自社の構成に読み替える
一次情報の確認日	2026年10月22日。本文中のURLは同日に直接確認した。出典一覧は末尾
本資料が示さないこと	30項目をすべて満たせば脆弱性がないという判断。本資料は自己点検の出発点であり、第三者の診断や監査の代わりにはならない
取扱い	自社内での複製・改変は自由。再配布時は出典を明記

この資料の使い方

本資料は、Webサービスを作っている開発者自身が、公開前や取引先への提出前に自分のコードと設定を見直すための点検表です。セキュリティ専任者がいない会社を前提に、「何を」「なぜ」「どこを見て」確認するかを30項目に絞っています。

誰が	いつ	どう使うか
開発者・テックリード	新規公開前、大きな機能追加のリリース前	30項目を順に確認し、末尾の印刷用チェックリストに記録する。「×」はチケット化する
開発責任者・情報システム担当	取引先からセキュリティチェックシートが届いたとき	点検の記録を回答の根拠にする。未確認の項目は「未確認」と書き、埋めない
受託開発会社のPM	納品前、保守契約の更新時	自己点検結果として納品物に添付する。発注元が求める水準との差分を見積の根拠にする
経営者	半年に1回	「×」と「未確認」の数の推移だけを見る

30項目は有限です。OWASP ASVS 5.0は17章・数百の要件からなり、本資料はそのうち、中小規模のWebサービスで見つかることが多く、開発者自身で確認できるものを選びました。WebRTC、決済代行との接続仕様、モバイルアプリ、社内ネットワークは扱いません。「30項目すべて○だから安全」とは言えず、本資料もそう主張しません。目的は、「×」や「未確認」が残っている状態に見えるようにし、直す順番を決めることです。項目番号（A-01～H-04）は領域ごとの通し番号で、重要度順ではありません。

なぜ今か

点検の必要性は、脅威の統計と、取引先・法令からの要求の両方から来ています。一次情報で確認できた事実だけを書きます。

事実	出典	確認日
IPA「情報セキュリティ10大脅威 2026」組織編で、「サプライチェーンや委託先を狙った攻撃」が2位（8年連続選出）、「システムの脆弱性を悪用した攻撃」が4位（9回目の選出）	IPA 情報セキュリティ10大脅威 2026（公開2026-01-29） https://www.ipa.go.jp/security/10threats/10threats2026.html	2026-10-22
IPA「2024年度 中小企業における情報セキュリティ対策に関する実態調査」（Webアンケート4,191社）で、不正アクセス被害を受けた企業（n=419）の手口は「脆弱性を悪用」が48.0%で最多。被害経験企業（n=975）の被害額は平均73万円、平均復旧日数5.8日。被害企業の約7割が取引先にも影響。セキュリティ専門部署がある企業は9.3%	IPA プレスリリース 2025-02-14 https://www.ipa.go.jp/pressrelease/2024/press20250214.html 、2025-05-27 https://www.ipa.go.jp/pressrelease/2025/press20250527.html	2026-10-22
クレジットカード・セキュリティガイドライン 6.0版（2025年3月）で、EC加盟店のシステムとWebサイトの「脆弱性対策」が指針対策になり、2025年4月以降は既契約加盟店も対象。内容に「脆弱性診断又はペネトレーションテストを定期的に実施し、必要な修正対応を行う」「開発またはカスタマイズしている場合はセキュアコーディングの確認、ソースコードレビュー、入力値チェック」「管理画面のアクセス制限、多要素認証、ログイン失敗時のアカウントロック」を含む。カード情報を非保持化した加盟店も対象（FAQ No.36）、規模・業種を問わない（FAQ No.73）。外部委託する場合は委託先が「脆弱性対策」を理解して構築・運用することを加盟店が求める。6.1版（2026年3月）は分かりやすさの改訂で、対策の追加はない	日本クレジット協会 6.1版 https://www.j-credit.or.jp/security/pdf/Creditcardsecurityguidelines_6.1_published.pdf 、6.0版FAQ https://www.j-credit.or.jp/security/pdf/creditcardsecurityguidelines_6.0_FAQ.pdf	2026-10-22
不正アクセスなど不正の目的によるおそれがある個人データの漏えい等は、件数にかかわらず個人情報保護委員会への報告対象。クレジットカード番号や決済機能のあるサービスのログイン情報の漏えいも対象。速報は発覚から概ね3～5日以内、確報は30日以内（不正目的のおそれがある場合は60日以内）	個人情報保護委員会「漏えい等の対応とお役立ち資料」 https://www.ppc.go.jp/personalinfo/legal/leakAction/	2026-10-22
2026年7月に成立した改正個人情報保護法で、重大な違反に対する課徴金制度が導入される。施行日は公布から2年以内で未定	個人情報保護委員会「令和8年 改正個人情報保護法について」 https://www.ppc.go.jp/personalinfo/legal/r8kai seihogohou	2026-10-22

数字が示すのは、脆弱性の悪用が「新しい脅威」ではなく「毎年同じ場所にある脅威」であること、被害が自社だけで終わらないことです。ガイドラインは診断を法律で義務付けてはいませんが、EC加盟店とその委託先に説明責任を置いています。自己点検はその説明責任の最初の一步です。第三者診断の前に自分で確認しておけば、指摘件数が減り、診断の時間を「ツールでは見つからない問題」に使えます。

30項目のチェック表

各項目の「確認方法」は、コードのどこを見るか、どのコマンド・ツールで補助できるかを書いています。ツールは無料で使えるオープンソースまたは各プラットフォームの標準機能を中心に挙げました。「参照」の略称は末尾の出典一覧に対応します。ASVSの要件番号はすべて5.0.0のものです（4.0.xとは番号が一致しません）。

「結果」欄には○（確認した・問題なし）、×（問題あり）、－（該当機能なし）、？（未確認）のいずれかを書きます。「？」のまま放置せず、確認できない理由を記録します。

認証・セッション（5項目）

No.	確認すること	なぜ危ないか	確認方法（コードのどこを見るか・ツール例）	参照	結果
A-01	パスワードを平文・可逆暗号・MD5／SHA-1の単純ハッシュで保存していない。bcrypt、scrypt、Argon2、PBKDF2のいずれかを使っている	DBが漏えいした時点で全パスワードが復元され、他サービスへの使い回し攻撃に転用される	登録・ログイン処理のハッシュ関数を読む。 <code>grep -rE "md5 sha1"</code> 。Semgrep <code>p/owasp-top-ten</code>	ASVS V6、CS 認証情報の保存	
A-02	ログイン・パスワードリセット・ワンタイムコードに試行回数制限（アカウント・IP単位）がある。失敗メッセージからアカウントの有無が分からない	漏えいパスワードの総当たりが成立する。クレカGLは管理画面について10回以下の失敗でロックを求める	ログインとリセットの処理に失敗回数のカウント・レート制限があるか。ステージングで11回失敗させる	ASVS V6、CS クレデンシャルスタッフィング、クレカGL	
A-03	管理画面・管理者アカウントに多要素認証がある。管理画面はIP制限または別認証で一般利用者から隔離されている	管理者1人のパスワード漏えいが全顧客データの漏えいに直結する	管理者ロールのログインフローにMFA検証があるか。 <code>/admin</code> 等のルートの前段ミドルウェア（IP制限、Basic認証）	ASVS V6、CS 多要素認証、クレカGL	
A-04	セッションIDはログイン成功時に再生成され、ログアウト・パスワード変更で無効化される。CookieにSecure、HttpOnly、SameSiteが付き、有効期限がある	セッション固定、盗まれたセッションの継続利用、XSS経由の窃取が成立する	ログイン成功時の <code>session.regenerate()</code> 相当。ログアウト時のサーバー側破棄。 <code>curl -sI</code> でSet-Cookieの属性。ZAP受動スキャン	ASVS V7（7.4.1、7.4.2）、V3（3.3.1）、CS セッション管理	
A-05	JWT・APIトークンはサーバー側で署名・有効期限・aud／issを検証し、アルゴリズムを固定している。固定APIキーで動く内部エンドポイントは棚卸しされ、ローテーション手順がある	検証漏れのトークンは改ざんで認証を素通りできる。固定キーは漏えいに気付かず、無効化もできない	<code>jwt.verify</code> 相当に <code>algorithm</code> ・ <code>audience</code> の指定があるか。 <code>grep -rE "x-api-key API_KEY"</code> で固定キーを列挙。改ざんトークンで401を確認	ASVS V7（7.2.1、7.2.2）、V9（9.1.1、9.2.1）、API2:2023	

認可・テナント分離（4項目）

No.	確認すること	なぜ危ないか	確認方法（コードのどこを見るか・ツール例）	参照	結果
B-01	リソースIDを受け取るすべてのエンドポイントで、呼び出し元がその所有者であることをサーバー側で確認している	IDを書き換えるだけで他人の注文・請求書・個人情報が読める（IDOR／BOLA）。API Security Top 10の1位で、自動スキャンでは見つからない	ルート一覧に「IDを受け取る」「所有者確認あり」「テスト済み」の列を作る。利用者Aのセッションで利用者BのIDを指定し403／404を確認	ASVS V8（8.2.2、8.3.1）、API1:2023、CS IDOR防止	

No.	確認すること	なぜ危ないか	確認方法（コードのどこを見るか・ツール例）	参照	結果
B-02	管理者向け機能（設定変更、全件エクスポート、ユーザー管理、課金変更）が一般ユーザーや未認証で呼べない。UI で隠しているだけになっていない	ボタンを隠しても API は呼べる。権限昇格で全データに到達する	ルートごとに必要ロールを列挙し、一般権限のセッションで管理 API を curl で直接呼ぶ。ロール確認がミドルウェアで一括か、ハンドラ任せか	ASVS V8 (8.2.1)、API5:2023、CS 認可	
B-03	更新 API がリクエストボディをそのまま保存していない（マスアサインメント）。応答に不要なフィールド（内部 ID、他ユーザー情報、トークン）を含めない	role: admin や price: 0 をボディに足すだけで権限や金額が書き換わる	...req.body や data: body を ORM に渡す箇所を grep。更新は許可フィールドの明示リストか。応答は DTO で項目を絞っているか	ASVS V15 (15.3.3)、V8 (8.2.3)、API3:2023、CS マスアサインメント	
B-04	マルチテナント構成では、すべてのデータアクセスにテナント ID の条件が入る。キャッシュキー、ストレージのパス、検索インデックス、バッチ処理も分離されている	テナント境界の欠落は1件で全顧客に波及し、SaaS の信頼を失う最大の要因になる	テナント条件を自動付与する仕組み（ORM スコープ、共通リポジトリ層、行レベルセキュリティ）があるか。生クエリ・集計・管理バッチの抜けを Semgrep カスタムルールで探す	ASVS V8 (8.4.1)、API1:2023	

入力検証・インジェクション（4項目）

No.	確認すること	なぜ危ないか	確認方法（コードのどこを見るか・ツール例）	参照	結果
C-01	DB へのクエリはすべてパラメータ化されている。文字列連結で SQL・NoSQL の条件を組み立てていない。NoSQL の演算子注入（\$gt、\$where）を防いでいる	SQL インジェクションは最も実害の大きい脆弱性であり、DB 全件の漏えい・改ざんに至る	queryRawUnsafe、executeRawUnsafe、文字列連結の SELECT / WHERE を grep し、raw クエリは全件読む。Semgrep、ZAP 能動スキャン（ステージング限定）	ASVS V1 (1.2.4)、CS SQL インジェクション防止	
C-02	すべての入力（ボディ、クエリ、パス、ヘッダー、Webhook）にサーバー側でスキーマ検証（型・長さ・許可値）がある	想定外の型や長さが、ロジックの迂回やインジェクションの入口になる	バリデーション（zod、Joi、JSON Schema、Pydantic 等）の適用をルートごとに一覧化。検証なしで req.body を使うハンドラを探す	ASVS V2 (2.2.1、2.2.2)、CS 入力値検証	
C-03	HTML 出力はテンプレートの自動エスケープに任せ、innerHTML、dangerouslySetInnerHTML、v-html 等の生出力は理由とサニタイズが明記されている。CSP を設定している	XSS はセッション窃取、管理者操作の乗っ取り、EC ではカード入力フォームの改ざんに使われる	grep -rE "innerHTML\s* = dangerouslySetInnerHTML v-html mark_safe"。curl -sI で CSP の有無。ZAP 受動スキャン	ASVS V1 (1.2.1、1.2.3)、V3 (3.2.2、3.4.6)、CS XSS 防止	
C-04	利用者指定の URL にサーバーがアクセスする機能（Webhook 通知先、画像取得、インポート）に、スキーム・ホストの許可リストと内部ネットワーク・メタデータエンドポイント（169.254.169.254）への到達防止がある。アップロードは拡張子・内容・サイズを検証する	SSRF はクラウドでは認証情報の窃取に直結する。アップロードの不備は任意コード実行やクレカGLの言う「データディレクトリの露見」につながる	fetch(、axios.、requests.get(の引数に外部入力が入る箇所と検証関数。ステージングで http://169.254.169.254/ を指定して拒否を確認	ASVS V1 (1.3.6)、V13 (13.2.4)、V5、API7:2023、CS SSRF 防止、CS ファイルアップロード	

秘密情報・設定（4項目）

No.	確認すること	なぜ危ないか	確認方法（コードのどこを見るか・ツール例）	参照	結果
D-01	リポジトリの現在のツリーと全履歴に、API キー、クラウド認証情報、DB 接続文字列、秘密鍵、 <code>.env</code> が含まれていない。含まれていた場合はローテーション済み	履歴に残った秘密情報は削除しても取り出せる。公開リポジトリでは数分で悪用される	<code>gitLeaks git -v --redact .</code> で全履歴をスキャン。GitHub の Secret scanning・Push protection の有効化（プランで利用条件が異なる）	ASVS V13（13.3.1、13.4.1）、CS シークレット管理、GitHub Push protection	
D-02	本番の秘密情報はシークレット管理基盤にあり、実行時に取得している。参照権限はサービスごとに最小。ローテーション手順がある	平文の環境変数や設定ファイルは、ログ・エラー画面・スナップショット経由で漏れる	デプロイ設定（IaC、コンテナ定義、PaaS の環境変数）から秘密情報の置き場所と読める主体を一覧化。最後にローテーションした日を記録	ASVS V13（13.3.1、13.3.2）、CS シークレット管理	
D-03	本番でデバッグモードが無効。エラー応答にスタックトレース・SQL・内部パスが出ない。 <code>.git/</code> 、 <code>.env</code> 、バックアップ、テスト用ルートが外部から見えない	内部構造の露出は攻撃の下調べを容易にする。 <code>.git</code> の公開はソースコード全体の漏えいと同じ	<code>curl -s https://<host>/.git/HEAD</code> 、 <code>/.env</code> 、存在しないパスへの応答本文。 <code>NODE_ENV</code> 、 <code>DEBUG</code> 、 <code>APP_DEBUG</code> の本番値。ZAP 受動スキャン	ASVS V13（13.4.1、13.4.2）、V16（16.5.1）、API8:2023、CS エラー処理	
D-04	全エンドポイントが HTTPS のみで HSTS を送る。TLS 1.2 以上。CORS の <code>Allow-Origin</code> は許可リストで <code>Origin</code> を反射しない。 <code>nosniff</code> 、 <code>Referrer-Policy</code> 、 <code>frame-ancestors</code> を送る	平文通信は盗聴・改ざんを許す。CORS の反射は他サイトから認証付きリクエストを許し、クリックジャッキングは管理操作の乗っ取りに使われる	<code>curl -sI https://<host>/</code> <code>curl -sI -H "Origin: http://evil.example" .../api/</code> で CORS 応答。 <code>testssl.sh</code> 、MDN HTTP Observatory	ASVS V12（12.1.1、12.2.1）、V3（3.4.1～3.4.6）、CS HTTP ヘッダー、CS TLS	

依存パッケージ・サプライチェーン（3項目）

No.	確認すること	なぜ危ないか	確認方法（コードのどこを見るか・ツール例）	参照	結果
E-01	依存パッケージの既知脆弱性を定期的に検出し、High 以上は到達可能性を判断して期限内に更新する。ロックファイルがコミットされ、CI は再現可能なインストール（ <code>npm ci</code> 等）。コンテナのベースイメージも対象	中小企業の不正アクセス被害の48.0%が脆弱性の悪用。多くは修正版が公開済みの脆弱性の放置	<code>npm audit --omit=dev --audit-level=high</code> 、OSV-Scanner、Trivy。Dependabot アラートの有効化と、未対応件数・経過日数	ASVS V15（15.2.1、15.1.2）、CS 脆弱な依存関係の管理、Dependabot	
E-02	CI/CD に、既定ブランチの保護、サードパーティアクションのコミット SHA 固定、ワークフローの最小権限、長期アクセスキーではなく OIDC によるクラウド認証、本番デプロイの承認ゲートがある	ビルド・デプロイ経路はアプリ本体より守りが薄く、1つの侵害で本番の全権限に到達する	<code>.github/workflows/*.yaml</code> （または同等）で <code>uses:</code> のタグ参照、 <code>permissions:</code> の明示、シークレットに残るクラウドのアクセスキーを確認。ブランチ保護と Environment の設定画面	GitHub Actions セキュリティ強化、CS CI/CD セキュリティ	
E-03	外部 API・SaaS の応答を検証してから保存・表示する。HTTP クライアントのリダイレクト自動追従を制御している。プライベートレジストリと公開レジストリの名前の取り違え（ディペンデンシーコンフュージョン）が起きない設定	外部サービスは自社の管理外で侵害されうる。「信頼できる相手からの入力」も入力である	外部応答を扱う箇所のスキーマ検証。 <code>maxRedirects</code> / <code>redirect: 'manual'</code> 。 <code>.npmrc</code> 等のレジストリとスコープの対応	ASVS V15（15.3.2）、V2（2.2.1）、API10:2023	

ログ・監視（3項目）

No.	確認すること	なぜ危ないか	確認方法（コードのどこを見るか・ツール例）	参照	結果
F-01	認証失敗、認可失敗、管理操作、権限変更がログに残る。各ログに時刻（UTC またはオフセット付き）、リクエスト ID、操作主体、送信元 IP が含まれる	ログがなければ侵害に気付かず、気付いても範囲を特定できない。漏えい報告には「何が漏れたか」の特定が要る	認証・認可の失敗パスに出力があるか。ログ1行のサンプルで項目の有無を確認。集約先と保持期間	ASVS V16（16.2.1、16.3.1、16.3.2）、CS ログ記録	
F-02	ログにパスワード、トークン、カード番号、個人情報、Webhook 本文全体が出ていない。閲覧権限が制限され、保持期間が決まっている	ログは「二次的な個人情報データベース」になり、漏えい報告の対象を自ら増やす	<code>console.log(JSON.stringify(req))</code> のように入力全体を出す箇所。マスク処理。ログ基盤のアクセス権限	ASVS V16（16.2.5、16.4.1、16.4.2）、CS ログ記録	
F-03	異常（認証失敗の急増、5xx の増加、クラウドの脅威検出の通知）が誰にどの経路で届くか決まっている。初動の判断者と、個人情報漏えい時の報告期限を担当者が知っている	検出しても誰も見ていなければ存在しないのと同じ。初動の遅れは報告期限の超過と被害拡大につながる	通知先の設定と、最後に通知が来て対応した記録。脅威検出サービス（AWS GuardDuty 等）の有効化。連絡先と報告手順が1枚の文書になっているか	ASVS V16、個人情報保護委員会 漏えい等報告	

データ保護・個人情報（3項目）

No.	確認すること	なぜ危ないか	確認方法（コードのどこを見るか・ツール例）	参照	結果
G-01	扱う個人情報の種類と保存場所（DB、ストレージ、ログ、バックアップ、外部 SaaS）の一覧がある。保持期間と削除手順が決まっている	何がどこにあるか分からなければ、漏えい時に報告対象かどうか判断できない。カード番号や決済機能付きサービスのログイン情報は1件でも報告対象になりうる	DB スキーマ・ストレージ・ログ・外部連携先から個人情報を含む列・パス・サービスを表にする。退会処理と、バックアップ・ログからの削除方針	ASVS V14、個人情報保護委員会 漏えい等報告	
G-02	DB・ストレージ・バックアップの保存時暗号化が有効。トークンや個人情報を URL のクエリ文字列で送受信していない。推測されはならない値に暗号的に安全な乱数を使っている	クエリ文字列はアクセスログ・Referer・履歴に残る。 <code>Math.random()</code> 相当の値は予測でき、リセットコードの推測に使われる	暗号化設定の実際の値。 <code>grep -r nE "\?.*(token email key) ="</code> 。 <code>Math.random()</code> / <code>random.random()</code> の用途を確認し <code>crypto.randomBytes</code> / <code>secrets</code> に置き換える	ASVS V14（14.2.1）、V11（11.3.1、11.4.1）、CS 暗号化保存	
G-03	本番の個人情報を開発環境やローカル端末にコピーしていない（必要ならマスク）。一括エクスポートは権限を限定し実行を記録する。機微な応答に <code>Cache-Control: no-store</code> 。 <code>localStorage</code> に認証情報を置かない。カード番号は自社で保持せず決済代行のトークンを使う	非本番環境は守りが薄く漏えいの起点になる。エクスポートは内部不正の主要経路。カード番号の自社保持は PCI DSS の対象範囲を広げる	開発環境 DB の中身。エクスポート API のロール要件と監査ログ。 <code>curl -sI</code> でキャッシュヘッダー。 <code>localStorage.setItem</code> の中身	ASVS V14（14.3.2、14.3.3）、クレカ GL	

クラウド設定の基本（4項目）

クラウドが AWS、Google Cloud、Azure、各種 PaaS のいずれでも、確認する観点は共通です。具体的な設定名は自社のクラウドのドキュメントに読み替えてください。

No.	確認すること	なぜ危ないか	確認方法（コードのどこを見るか・ツール例）	参照	結果
H-01	ルート／オーナー権限のアカウントに MFA があり、日常作業に使っていない。人間の利用者は個人ごとのアカウントで MFA 必須。退職者・契約終了した委託先のアカウントと長期アクセスキーが残っていない	最上位アカウントの乗っ取りは全資産の削除・持ち出しに直結する。退職者のキーは最も見落とされやすい侵入経路	利用者一覧に最終ログイン日・MFA の有無・アクセスキーの作成日を付けて表にする（AWS では IAM 認証情報レポート）。Prowler や各クラウド標準の設定チェック	AWS IAM ベストプラクティス、CIS Benchmarks、Prowler	
H-02	アプリ・CI・運用者の権限が最小限。管理者相当の権限を持つ主体の数を把握している。ポリシーの*（全リソース・全アクション）には理由が記録されている	過剰権限は、アプリの1つの脆弱性（SSRF、依存パッケージの侵害）をアカウント全体の侵害に広げる	管理者権限が付いた主体を列挙。アプリ実行ロールのポリシー本文を読む。未使用権限の分析機能（AWS IAM Access Analyzer 等）	AWS IAM ベストプラクティス、CS セキュアなクラウドアーキテクチャ	
H-03	ストレージのバケット、DB、キャッシュ、管理ポート（SSH、RDP、DB ポート）がインターネットに公開されていない。公開が必要なものは一覧化され理由がある	公開バケットと公開 DB は、攻撃ではなく「検索」で見つかる。設定ミス1つで全件漏えいになる	パブリックアクセスブロック、ファイアウォールの 0.0.0.0/0 許可を一覧化。Prowler、AWS Security Hub 等の結果から「公開」に関する検出を先に見る	CIS Benchmarks、Prowler	
H-04	クラウドの操作ログ（AWS CloudTrail 等）が全リージョンで有効で、改ざんできない場所に保管されている。請求額のアラートがある。バックアップは自動で、復元を試した記録がある	操作ログがなければ侵害の経路が分からない。請求アラートは不正利用の最も早い検知手段になることが多い。試していないバックアップは存在しないのと同じ	監査ログの設定と保存先の権限。請求アラートの閾値と通知先。バックアップの最終取得日と復元テストの日	CIS Benchmarks、CS セキュアなクラウドアーキテクチャ	

優先順位の付け方

「×」が10個付いたとき、上から順に直すのは正しくありません。CVSSの数値だけで並べるのも、自社の構成で成立しない指摘が上位に来るので勧めません。Forest Securityの診断で使っているのと同じ3軸で判定します。

軸	問い	高い例	低い例
攻撃の成立しやすさ	認証なしで外部から成立するか。特別な条件が要るか	未認証で B-01 (IDOR) が成立する。D-01 で公開リポジトリに秘密情報	管理者権限を持つ内部者だけが悪用できる
影響を受ける資産	到達できるのは何か	全顧客の個人情報、決済情報、クラウドの管理権限、他テナントのデータ	公開情報、内部構成の一部
業務影響	成立したとき事業に何が起きるか	大量漏えいによる報告義務・取引停止、他テナントへの波及、サービス停止	防御の多層性が1枚薄い

判定の手順は次のとおりです。

- 3軸それぞれの水準を決め、最も高い水準を基本の重大度とする
- 「攻撃の成立しやすさ」が低く「影響を受ける資産」だけが高い場合は、1段下げる（成立しにくいものを最上位にしない）
- 同じ重大度なら、修正に要する時間が短いものを先にする。設計変更が必要な項目は着手日だけ先に決め、他を止めない
- 「？（未確認）」は「×」と同じ扱いで期限を付ける。確認できない理由（権限がない、作った人がいない）自体が指摘である

一般に、30項目の中で最初に見るべきなのは、D-01（履歴内の秘密情報）、H-01（最上位アカウントと退職者のキー）、H-03（公開設定）、B-01（IDOR）、C-01（インジェクション）です。これらは「外部から、認証なしで、全件に到達する」経路になりやすいためです。ただしこれは一般論であり、自社の構成で3軸を当てはめた結果を優先してください。

自己点検の後にやること

点検は記録して初めて意味を持ちます。点検の後は次の順に進めます。

段階	内容	記録に残すもの
チケット化	「×」「？」を1項目1チケットにする。項目番号（A-01等）、確認した日、確認者、該当箇所（ファイル・設定名）、再現条件を書く	チケット一覧
直す順番の合意	3軸で重大度を付け、修正規模を加味して着手順を決める。担当は個人名	重大度、担当、期限
修正	チケットに沿って修正する。PRに項目番号を書く	PRのURL
再確認	修正者以外の人、修正前に成立した条件が成立しないことを確かめる。「確認した範囲」（どの環境で、どの手順で）を書く	再確認記録（確認者、日付、範囲、方法）

段階	内容	記録に残すもの
記録	印刷用チェックリストを更新し、点検日・版・残件数を保存する。取引先への回答は、この記録を根拠にする	点検記録の保管先
次回の予定	大きなリリース前と、半年ごとに再点検する日を決める。依存パッケージ（E-01）とクラウドの公開設定（H-03）は月1回の頻度で見る	次回点検日

再確認を修正者自身が行う運用では、ステージングにだけ修正が入っていた、環境変数は消したが IaC のテンプレートに残っていた、といった取りこぼしが残ります。再確認は別の人が行います。

次の場面では、自己点検だけでは足りず、第三者の確認が必要になります。

場面	理由
取引先・発注元から「第三者による診断結果」の提出を求められた	自己点検は当事者の記録であり、第三者の確認には当たらない
審査・申請で第三者の確認が条件になっている（プラットフォームのデータ保護審査、業界ガイドラインへの対応状況の申告等）	審査側が求める根拠の形式が決まっている
B-01、B-04、C-01 を自分たちで確認し切れない（エンドポイント数が多い、作った人がいない）	認可とテナント分離は自動ツールで見つからず、第三者の目が効く領域
「？」が5個以上残り、確認できる人が社内にはいない	確認できない理由の多くは、権限・知識・時間のいずれかの不足であり、外部に頼む方が早い
インシデントの疑いがある	自己点検の範囲外。専門のインシデント対応事業者を引き継ぐ

第三者に依頼するときは、本資料の点検結果をそのまま渡してください。確認済みの項目を飛ばし、未確認・不安な項目に診断の時間を集中できます。

印刷用チェックリスト

結果の記号：○ 確認した・問題なし / × 問題あり / - 該当機能なし / ? 未確認（確認できない理由を欄外に書く）。

点検対象	点検日	点検者	対象の版・コミット

No.	項目	結果	担当	期限	No.	項目	結果	担当	期限
A-01	パスワードの保存方式				D-03	デバッグ・エラー応答・不要公開			
A-02	ログイン試行の制限				D-04	TLS・CORS・セキュリティヘッダー			
A-03	管理画面の MFA と隔離				E-01	依存パッケージの既知脆弱性			
A-04	セッション管理と Cookie 属性				E-02	CI/CD の保護			
A-05	トークン検証・固定キー				E-03	外部応答の検証・取得元			
B-01	オブジェクト単位の認可				F-01	セキュリティイベントのログ			
B-02	機能単位の認可				F-02	ログへの機微情報			
B-03	プロパティ単位の認可				F-03	通知先と初動			
B-04	テナント分離				G-01	個人情報の棚卸し			
C-01	クエリのパラメータ化				G-02	保存時暗号化・URL・乱数			
C-02	サーバー側の入力検証				G-03	非本番環境・エクスポート・保持			
C-03	出力エンコードと CSP				H-01	最上位アカウントと利用者の MFA			
C-04	SSRF・アップロード				H-02	最小権限			
D-01	履歴内の秘密情報				H-03	公開設定			
D-02	秘密情報の保管とローテーション				H-04	操作ログ・請求アラート・バックアップ			

集計：○ 件 / × 件 / - 件 / ? 件

Forest Securityについて

Forest Securityは、株式会社Forest Bookが提供する、Webサービス（Webアプリ、API、SaaS、EC、会員サイト、社内システムの外部連携）を自社で開発・運用する中小企業・開発会社向けのセキュリティ診断と是正管理のサービスです。ソースコードとクラウド設定の両方から問題を確認し、直す順番を決め、直るまでの進捗を管理します。対象はWebサービス全般で、クラウドはAWS・Google Cloud・Azure等を扱います。

強みは、Shopifyアプリ（公開・カスタム）とAWSサーバーレス（Lambda、API Gateway、DynamoDB、SQS、EventBridge、S3、Step Functions）を自分で開発してきた経験にあります。この領域は確認の密度と速度が上がりますが、対象をこの2つに限定しません。Google Cloud・Azure等は構成を伺ったうえで範囲を決めます。

本資料の30項目は、Forest Securityの診断で使うチェックリスト（OWASP ASVS 5.0.0、OWASP API Security Top 10 2023、クラウド各社の公式ドキュメントに対応付けたもの）から、開発者自身で確認できるものを抜き出したものです。診断では、認可・テナント分離の全エンドポイント確認、動的検査、読み取り専用権限によるクラウド設定の点検、是正後の再確認を行います。

商品	内容
Forest Security Review	Webアプリ／API 1本、またはクラウドアカウント1つを対象に、指摘一覧（ID、証拠、影響、重大度、推奨修正）と60分の報告会を納める。資料・権限受領から5営業日
Forest Security Assessment	Webアプリ／API 1本+クラウドアカウント1つを対象に、Web/API、クラウド設定、CI/CD・サプライチェーン、依存パッケージの4領域を確認し、技術報告書、リスク台帳、修正チケット案を納める。是正後の再確認1回を含む
Forest Security Care	月額で、クラウド設定の差分、依存パッケージ、脅威検出の結果のレビューと是正チケットの進捗管理を行い、月次報告書を出す

やらないこと：24時間365日の監視・SOC・MDR、インシデント発生時の復旧作業・フォレンジック・監督官庁への届出代行、網羅的なペネトレーションテスト、負荷試験、ソーシャルエンジニアリング、SaaS・PaaS本体（Shopify、決済代行会社、CDN等）や第三者が運営する資産への検査、脆弱性が存在しないことの保証、法令・ガイドライン・認証への適合や審査・申請の可否の保証、自社（Forest Book）が開発・修正した資産に対する独立した第三者としての監査意見、無制限の修正実装（修正はスポット支援として時間単位で別途提供）。

料金は対象資産の規模により個別に見積もります。相談は無料（30分）です。

問い合わせ先：株式会社Forest Book info@forest-book.jp

サービスサイト：<https://security.forest-book.jp>

〒220-0004 神奈川県横浜市西区北幸二丁目10番48号 むつみビル3階

出典一覧

統計・法令・ガイドライン（いずれも2026-10-22確認）

文書	URL
IPA「情報セキュリティ10大脅威 2026」（公開2026-01-29）	https://www.ipa.go.jp/security/10threats/10threats2026.html
IPA「2024年度 中小企業における情報セキュリティ対策に関する実態調査」速報（2025-02-14）	https://www.ipa.go.jp/pressrelease/2024/press20250214.html
同 報告書プレスリリース（2025-05-27）	https://www.ipa.go.jp/pressrelease/2025/press20250527.html
IPA「安全なウェブサイトの作り方」	https://www.ipa.go.jp/security/vuln/websecurity/about.html
日本クレジット協会「クレジットカード・セキュリティガイドライン」6.1版（クレカGL）	https://www.j-credit.or.jp/security/pdf/Creditcardsecurityguidelines_6.1_published.pdf
同 6.0版 FAQ	https://www.j-credit.or.jp/security/pdf/creditcardsecurityguidelines_6.0_FAQ.pdf
個人情報保護委員会「漏えい等の対応とお役立ち資料」	https://www.ppc.go.jp/personalinfo/legal/leakAction/
個人情報保護委員会「令和8年 改正個人情報保護法について」	https://www.ppc.go.jp/personalinfo/legal/r8kaiseihogohou

標準（いずれも2026-10-22確認）

略称	文書	URL
ASVS	OWASP Application Security Verification Standard 5.0.0（2025年5月）。要件番号は5.0.0のもの	https://github.com/OWASP/ASVS
API1～API10:2023	OWASP API Security Top 10 2023	https://owasp.org/API-Security/editions/2023/en/0x11-t10/
OWASP Top 10	OWASP Top 10	https://owasp.org/Top10/
CIS Benchmarks	Center for Internet Security Benchmarks（クラウド各社向けの設定基準）	https://www.cisecurity.org/cis-benchmarks
AWS IAM ベストプラクティス	AWS「Security best practices in IAM」	https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html
GitHub Actions セキュリティ強化	GitHub Docs「Security hardening for GitHub Actions」	https://docs.github.com/en/actions/security-for-github-actions/security-guides/security-hardening-for-github-actions

略称	文書	URL
GitHub Push protection	GitHub Docs 「About push protection」	https://docs.github.com/en/code-security/secret-scanning/introduction/about-push-protection
Dependabot	GitHub Docs 「About Dependabot security updates」	https://docs.github.com/en/code-security/dependabot/dependabot-security-updates/about-dependabot-security-updates

OWASP Cheat Sheet Series（本文中「CS」。いずれも2026-10-22確認）

略称	URL
CS 認証情報の保存	https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html
CS クレデンシャルスタッフィング	https://cheatsheetseries.owasp.org/cheatsheets/Credential_Stuffing_Prevention_Cheat_Sheet.html
CS パスワード忘れ	https://cheatsheetseries.owasp.org/cheatsheets/Forgot_Password_Cheat_Sheet.html
CS 多要素認証	https://cheatsheetseries.owasp.org/cheatsheets/Multifactor_Authentication_Cheat_Sheet.html
CS セッション管理	https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html
CS 認可	https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html
CS IDOR 防止	https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html
CS マスアサインメント	https://cheatsheetseries.owasp.org/cheatsheets/Mass_Assignment_Cheat_Sheet.html
CS SQL インジェクション防止	https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
CS 入力値検証	https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html
CS XSS 防止	https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html
CS SSRF 防止	https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html
CS ファイルアップロード	https://cheatsheetseries.owasp.org/cheatsheets/File_Upload_Cheat_Sheet.html
CS シークレット管理	https://cheatsheetseries.owasp.org/cheatsheets/Secrets_Management_Cheat_Sheet.html
CS エラー処理	https://cheatsheetseries.owasp.org/cheatsheets/Error_Handling_Cheat_Sheet.html
CS HTTP ヘッダー	https://cheatsheetseries.owasp.org/cheatsheets/HTTP-Headers_Cheat_Sheet.html
CS TLS	https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Security_Cheat_Sheet.html
CS 脆弱な依存関係の管理	https://cheatsheetseries.owasp.org/cheatsheets/Vulnerable_Dependency_Management_Cheat_Sheet.html
CS CI/CD セキュリティ	https://cheatsheetseries.owasp.org/cheatsheets/CI_CD_Security_Cheat_Sheet.html
CS ログ記録	https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html

略称	URL
CS 暗号化保存	https://cheatsheetseries.owasp.org/cheatsheets/Cryptographic_Storage_Cheat_Sheet.html
CS セキュアなクラウドアーキテクチャ	https://cheatsheetseries.owasp.org/cheatsheets/Secure_Cloud_Architecture_Cheat_Sheet.html

ツール（いずれも2026-10-22確認。名称の記載は例示であり、特定製品の推奨・保証ではない）

ツール	用途	URL
OWASP ZAP	Web アプリの受動・能動スキャン	https://www.zaproxy.org/
Semgrep	ソースコードの静的解析	https://semgrep.dev/
gitleaks	リポジトリ履歴の秘密情報検出	https://gitleaks.io/
npm audit	npm 依存パッケージの既知脆弱性検出	https://docs.npmjs.com/cli/v10/commands/npm-audit
OSV-Scanner	複数言語の依存パッケージの既知脆弱性検出	https://google.github.io/osv-scanner/
Trivy	コンテナイメージ・ファイルシステムの脆弱性検出	https://trivy.dev/
Prowler	クラウド設定の点検	https://docs.prowler.com/
testssl.sh	TLS 設定の確認	https://testssl.sh/
MDN HTTP Observatory	HTTP セキュリティヘッダーの採点	https://developer.mozilla.org/en-US/observatory

本資料の記載は2026年10月22日時点の公開情報に基づきます。ASVS の要件番号は GitHub リポジトリのタグ `v5.0.0_release` 配下の原文で確認しました。